Date=28/12/2020
Lecture By= Akash Handa
Subject ⇒ Project Day 1(M-tube)

IN PREVIOUS LECTURE (QUICK RECAP) Date-24/12/2020	**In Today's Lecture (Overview)**

M-tube Project
React-google-login
    Stay Logged in
    onSuccess callback
YouTube Data API Overview
    Introduction
        Before you start
    Resources and resource types
	Fetching most popular videos

In today's Lecture We Tried To implement Trending Videos On the home page Using Youtube Api
Lecture Link
http://d3w35yba4tosuh.cloudfront.net/ee9495b0-48ce-11eb-b7f2-35f055f313bb.mp4

Fetching most popular videos

we will load the top 12 most popular videos and all video categories from the Youtube API. Loading the most popular videos for a specific category works exactly like loading most popular videos. The only difference is that we need to pass a category id as a parameter in our request. You can also see what request we are about to do when you have a look at it in the API explorer.
There is one super strange thing here. I performed the following request:

```
{
  "kind": "youtube#videoListResponse",
  "nextPageToken": "CAUQAA",
  "pageInfo": {
   "totalResults": 200,
   "resultsPerPage": 5
  },
  "items": [
   {
    "kind": "youtube#video",
    "id": "2vWp2spX4FU",
    "snippet": {
     "publishedAt": "2018-10-22T16:12:23.000Z",
     "channelId": "UCKuHFYu3smtrl2AwwMOXOlg",
     "title": "Jada Finally Broke Me.",
     "channelTitle": "Will Smith",
     "categoryId": "24"
    },
    "contentDetails": "..."
   }
  ]
 }
```
GET
https://www.googleapis.com/youtube/v3/videos?part=snippet%2CcontentDetails%2Cstatistics&chart=mostPopular&videoCategoryId=1&key={YOUR_API_KEY}
I shortened the endpoint's response a little bit here so that the code section doesn't get too long.
Note that I explicitly asked for videos that are associated with video category id 1.
However, if you look at line 17 you see that the video apparently belongs to the category 24.
This is one strange thing about the Youtube API.

If you explicitly ask for videos from a specific video category, the returned videos might contain a different category id. This is somewhat strange. We need to work around that. So when we make the request, we must remember the category id for which we are requesting trending videos. Otherwise we might mess things up.

On a semantic level there's a simple explanation for this behaviour. A video can simply belong to multiple categories. However it is strange that the endpoint does not return the video category id we put in the request. Please keep that in mind for later.

Since we are already able to fetch the most popular videos, let's tweak our function a little bit so that we can restrict the returned videos to a specific category.

Head over to the `src/store/api/youtube-api` file and update:

```
export function buildMostPopularVideosRequest(amount = 12, loadDescription = false, nextPageToken, videoCategoryId = null) {
  let fields =
'nextPageToken,prevPageToken,items(contentDetails/duration,id,snippet(channelId,channelTitle,publishedAt,thumbnails/medium,title),statistics/viewCount),pageInfo(totalResults)';
  if (loadDescription) {
    fields += ',items/snippet/description';
  }
  return buildApiRequest('GET',
    '/youtube/v3/videos',
    {
      part: 'snippet,statistics,contentDetails',
      chart: 'mostPopular',
      maxResults: amount,
      regionCode: 'US',
      pageToken: nextPageToken,
      fields,
      videoCategoryId,
    }, null);
}
```

We added a new parameter called `videoCategoryId` and give it a default value of `null`. Remember that our `buildApiRequest` function will kick out all parameters where the value is `null`. So in case we don't explicitly specify a `video category id`, we will just fetch the most popular videos.

Before we move on and create the sagas and the reducer, we need to think about how we want to organise our state. We must somehow store which videos are associated with a certain category. In fact we can just recycle the approach we took when we included the most popular videos into our state.

```
{
  "api": {
      "libraryLoaded": false
   },
   "videos": {
      "byId": {
        "FLqvTE1Eqfg": {
         "kind": "youtube#video",
         "etag":
"\"XI7nbFXulYBIpL0ayR_gDh3eu1k/rzYqHJdz-a40clbPa3V5RJul7XU\"",
         "...": "..."
      },
      "mostPopular": {
        "items": ["FLqvTE1Eqfg", "..."],
        "totalResults": 200,
        "nextPageToken": "some-cryptic-token"
      },
      "categories": {
  "1": "Film & Animation",
  "2": "Music"
      },
      "byCategory": {
        "1": {
          "items": ["some-video-id", "..."],
          "totalResults": 182,
          "nextPageToken": "..."
        }
      }
   }
}
```

For Complete Guidance Follow the link below
https://productioncoder.com/build-youtube-in-react-part-29/